Spring 5-1-1990

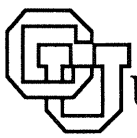
Connectionist Music Composition Based on Melodic, Stylistic, and Psychophysical Constraints ; CU-CS-495-90

Michael C. Mozer
University of Colorado Boulder

CONNECTIONIST MUSIC COMPOSITION BASED ON MELODIC, STYLISTIC, AND PSYCHOPHYSICAL CONSTRAINTS

Michael C. Mozer

CU-CS-495-90



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Abstract

Algorithmic music composition involves the use of rules to generate melodies. One simple but interesting technique is to select notes sequentially according to a transition table that specifies the probability of the next note as a function of the previous context. I describe an extension of this transition table approach using a recurrent connectionist network called CONCERT. CONCERT is trained on a set of melodies written in a certain style and then is able to compose new melodies in the same style. A central ingredient of CONCERT is the incorporation of a psychologically-grounded representation of pitch. CONCERT was tested on sets of examples artificially generated according to simple rules and was shown to learn the underlying structure, even where other approaches failed. In a larger experiment, CONCERT was trained on a set of J. S. Bach minuets and marches and was then allowed to compose novel melodies. Although the compositions are pleasant, I don't forsee a Grammy in the near future. The main problem is a lack of global coherence. Some ideas are presented about how a network can be made to induce structure at both local and global scales.

In creating music, composers bring to bear a wealth of knowledge of musical conventions. Some of this knowledge is based on the experience of the individual, some is culture specific, and perhaps some is universal. No matter what the source, this knowledge acts to constrain the composition process, specifying, for example, the musical pitches that form a scale, the pitch or chord progressions that are agreeable, and stylistic conventions like the division of a symphony into movements and the AABB form of a gavotte. If we hope to build automatic music composition systems that can mimic the abilities of a human composer, it will be necessary to incorporate knowledge of musical conventions into the systems. The difficulty is in deriving this knowledge in an explicit form: even human composers are unaware of many of the constraints under which they operate (Loy, in press).

In this chapter, I describe a connectionist network that composes melodies. The network is called CONCERT, an acronym for connectionist composer of erudite tunes. (The "er" may also be read as erratic or ersatz, depending on what the listener thinks of its creations.) Musical knowledge is incorporated into CONCERT via two routes. First, CONCERT is trained on a set of sample melodies from which it extracts rules of note and phrase progressions, which I call *melodic and stylistic constraints*. Second, I have built a representation of pitch into CONCERT, and have proposed an analogous representation of duration, that is based on psychological studies of human perception. This representation, and an associated theory of generalization proposed by Shepard (1987), provides CONCERT with a basis for judging the similarity among notes, for selecting a response, and for restricting the set of alternatives that can be considered at any time. I call these constraints imposed by the representation *psychophysical constraints*.

My experiments have been with single-voice melodies, most having 10–20 notes, but I also report on preliminary work with longer pieces having about 150 notes. A complete model of music composition should describe each note by a variety of properties — pitch, duration, phrasing, accent — along with more global properties such as tempo and dynamics. In most of the experiments reported here, the problem has been stripped to its bare bones, describing a melody simply as a sequence of pitches. Extending the work to notes that vary in duration or other properties is a relatively straightforward once the viability of the approach has been established. The burden of the present work has been to demonstrate that CONCERT can discover the appropriate structure in a set of pitch sequences presented to it.[1]

One potential pitfall in the research area of connectionist music composition is the uncritical acceptance of a network's performance. It is absolutely essential that a network be evaluated according to some objective criterion. One cannot judge the enterprise to be a success simply because the network is creating novel output. Even random note sequences played through a synthesizer sound interesting to many observers. Although Todd's (1989) seminal work on connectionist composition shows great promise, it suffers by the lack of evaluation; consequently, one cannot verify that his network architecture and learning algorithm have the computational power to succeed. In contrast, CONCERT is motivated by well-defined computational goals, which provide the means for evaluating its performance.

Another serious pitfall in connectionist research, and artificial intelligence research in general, is the assumption that if a system peforms well on small problems, it can readily be scaled up to handle larger problems of the same type. In the music composition domain, this assumption might lead to the tenuous belief that a network capable of composing ten-note melodies could, simply by adding more hidden units and connections, compose symphonies. A symphony, however, has structure at many levels: within a phrase, between phrases within a movement, between movements, etc. To discover rules of composition for notes within a phrase, only local temporal contingencies need to be examined. To

[1] Because my work to date considers primarily the pitch of a note, I use the terms "note" and "pitch" interchangeably.

discover rules of composition at a more global level, the network may need to examine the relationships among many *thousands* of notes. The sort of learning procedure used to discover local structure in music has no guarantee of success in discovering more global structure. A central focus of my work has been on this issue of dealing with structure at different levels. I have developed specialized network dynamics to address this issue.

Before turning to the details of my approach, I begin by describing a traditional approach to algorithmic music composition using Markov transition tables, the limitations of this approach, and how these limitations may be overcome using connectionist learning techniques.

Transition Table Approaches to Algorithmic Music Composition

One simple but interesting technique in algorithmic music composition is to select notes sequentially according to a *transition table* that specifies the probability of the next note as a function of the current note (Dodge & Jerse, 1985; Jones, 1981; Lorrain, 1980). For example, the transition probabilities depicted in Table 1 constrain the next pitch to be one step up or down the C major scale from the current pitch. Generating a sequence according to this probability distribution therefore results in a musical random walk. Transition tables may be hand-constructed according to certain criteria, as in Table 1, or they may be set up to embody a particular musical style. In the latter case, statistics are collected over a set of examples (hereafter, the *training set*) and the transition table entries are defined to be the transition probabilities in these examples.

The transition table is a statistical description of the training set. In most cases, the transition table will lose information about the training set. To illustrate, consider the two sequences `A B C` and `E F G`. The transition table constructed from these examples will indicate that `A` goes to `B` with probability 1, `B` to `C` with probability 1, and so forth. Consequently, given the first note of each sequence, the table can be used to recover the complete sequence. However, with two sequences like `B A C` and `D A E`, the transition table can only say that following an `A` either an `E` or a `C` occurs, each with a 50% likelihood. Thus, the table cannot be used to unambiguously reconstruct the examples.

Clearly, in melodies of any complexity, musical structure cannot be fully described by the pairwise statistics. To capture additional structure, the transition table can be generalized from a two-dimensional array to n dimensions. In the n-dimensional table, often referred to as a table of order $n-1$, the probability of the next note is indicated as a function of the previous $n-1$ notes. By increasing the number of previous notes taken into consideration, the table becomes more context sensitive, and therefore serves as a

Table 1: Transition probability from current pitch to the next

next pitch	*current pitch* C	D	E	F	G	A	B
C	0	.5	0	0	0	0	.5
D	.5	0	.5	0	0	0	0
E	0	.5	0	.5	0	0	0
F	0	0	.5	0	.5	0	0
G	0	0	0	.5	0	.5	0
A	0	0	0	0	.5	0	.5
B	.5	0	0	0	0	.5	0

more faithful representation of the training set.[2] Unfortunately, extending the transition table in this manner gives rise to two problems. First, the size of the table explodes exponentially with the amount of context and rapidly becomes unmanageable. With, say, 50 alternative notes and a third-order transition table — modest sizes on both counts — 6.25 million entries would be required. Second, a table representing the high-order structure masks the tremendous amount of low-order structure present. To elaborate, consider the sequence

A F G B F G C F G D F G# E F G .

One would need to construct a third-order table to faithfully represent this sequence. Such a table would indicate that, for example, the sequence G B F is always followed by G. However, there are first-order regularities in the sequence that a third-order table does not make explicit, namely the fact that an F is almost always followed by a G. The third-order table is thus unable to predict what will follow, say, A A F, although a first-order table would sensibly predict G. There is a tradeoff between the ability to faithfully represent the training set, which usually requires a high-order table, and the ability to generalize in novel contexts, which profits from a low-order table. What one would really like is a scheme by which only the *relevant* high-order structure is represented.

Kohonen (1989; Kohonen, Laine, Tiits, & Torkkola, in press) has proposed exactly such a scheme. The scheme is symbolic algorithm that, given a training set of examples, produces a collection of rules — a context-sensitive grammar — sufficient for reproducing most or all of the structure inherent in the set. These rules are of the form *context*→*next_note*, where *context* is a string of one or more notes, and *next_note* is the next note implied by the context. Because the context length can vary from one rule to the next, the algorithm allows for varying amounts of generality and specificity in the rules. The algorithm attempts to produce deterministic rules — rules that always apply in the given context. Thus, the algorithm will not discover the regularity F → G in the above sequence because it is not absolute. One could conceivably extend the algorithm to generate simple rules like F → G along with exceptions (e.g., D F → G#), but the symbolic nature of the algorithm still leaves it poorly equipped to deal with statistical properties of the data. Such an ability is not critical if the algorithm's goal is to construct a set of rules from which the training set can be exactly reconstructed. However, my view is that music composition is an intrinsically random process and it is therefore inappropriate to model every detail of the training set. Instead, the goal ought to be to capture the most important — i.e., statistically regular — structural properties of the training set.

Both the transition table approach and Kohonen's musical grammar suffer from two further drawbacks. First, both algorithms are designed so that a particular note, n, cannot be used to predict note $n+i$ unless all intervening notes, $n+1 \cdots n+i-1$, are also considered. In general, one would expect that the most useful predictor of a note is the immediately preceding note, but cases exist where notes $n \cdots n+k$ are more useful predictors of note $n+i$ than notes $n+k+1 \cdots n+i-1$ (e.g., a melody in which high pitch and low pitch phrases alternate such as the solo violin partitas of J. S. Bach). The second drawback is that a symbolic representation of notes does not facilitate generalization. For instance, invariance under transposition is not directly representable. In addition, other similarities are not encoded, for example, the congruity of octaves.

---

[2] Following Smolensky (1988), I use the phrase *faithful representation* to mean that the represented items can be accurately reconstructed from the representation. A faithful transition-table representation of a set of examples would be one that, given the first few notes of any example, could unambiguously determine the remainder of the example.

Connectionist learning algorithms offer the potential of overcoming the various limitations of transition table approaches and Kohonen musical grammars. Connectionist algorithms are able to discover relevant structure and statistical regularities in sequences (e.g., Elman, 1990; Mozer, 1989). Indeed, connectionist algorithms can be viewed as an extension of the transition table approach, a point also noted by Dolson (1989). Just as the transition table approach uses a training set to calculate the probability of the next note in a sequence as a function of the previous notes, so does CONCERT. The connectionist approach, however, is far more flexible: The form of the transition function can permit the consideration of varying amounts of context, the consideration of noncontiguous context, and the combination of low-order and high-order regularities.

The connectionist approach also promises better generalization through the use of distributed representations (Hinton, McClelland, & Rumelhart, 1986). In a local representation, where each note is represented by a discrete symbol, the sort of statistical contingencies that can be discovered are among notes. However, in a distributed representation, where each note is represented by a set of continuous feature values, the sort of contingencies that can be discovered are among *features*. To the extent that two notes share features, featural regularities discovered for one note may transfer to the other note.

The CONCERT Architecture

CONCERT is a recurrent network architecture of the sort studied by Elman (1990). A melody is presented to it, one note at a time, and its task at each point in time is to predict the next note in the melody. Using a training procedure described below, CONCERT's connection strengths are adjusted so that it can perform this task correctly for a set of training examples. Each example consists of a sequence of notes. The current note in the sequence is represented in the input layer of CONCERT, and the prediction of the next note is represented in the output layer. As Figure 1 indicates, the next note is encoded in two different ways: The next-note-distributed (or *NND*) layer contains CONCERT's internal representation of the note, while the next-note-local (or *NNL*) layer contains one unit for each alternative. The representation of a note in the NND layer, as well as in the input layer, is based on a psychological analysis of human pitch perception (Shepard, 1982), which I explain in detail in a following section. For now, it should suffice to say that this representation is distributed, i.e., a note is indicated by a *pattern* of activity across the units. Because such patterns of activity can be quite difficult to interpret, the NNL layer provides an alternative, explicit representation of the possibilities.

The context layer can represent relevant aspects of the input history, that is, the temporal context in which a prediction is made. When a new note is presented in the input layer, the activity pattern currently in the context layer is integrated with the new note to form a new context representation. In general terms,

$$\mathbf{c}(n) = f\,(\mathbf{c}(n-1), \mathbf{x}(n))\,,$$

where $\mathbf{x}(n)$ is a vector representing the nth note in the input sequence, $\mathbf{c}(n)$ is the context activity pattern following processing of input note n — which I refer to as *step* n — and f is a member of the class of functions that can be implemented by the connectionist hardware. At the start of each sequence the context layer is cleared, i.e., $\mathbf{c}(0) = \mathbf{0}$.

CONCERT could readily be wired up to behave as a k-th order transition table. In this case, the function f is defined to implement a k element stack in the context layer. This stack would hold on to notes $n-k+1$ through n. The connections from the context layer to the output layer would then have to be set up to realize a look-up table in which each combination of previous notes maps to the appropriate probability distribution over the next note. However, the architecture is more general than a transition table

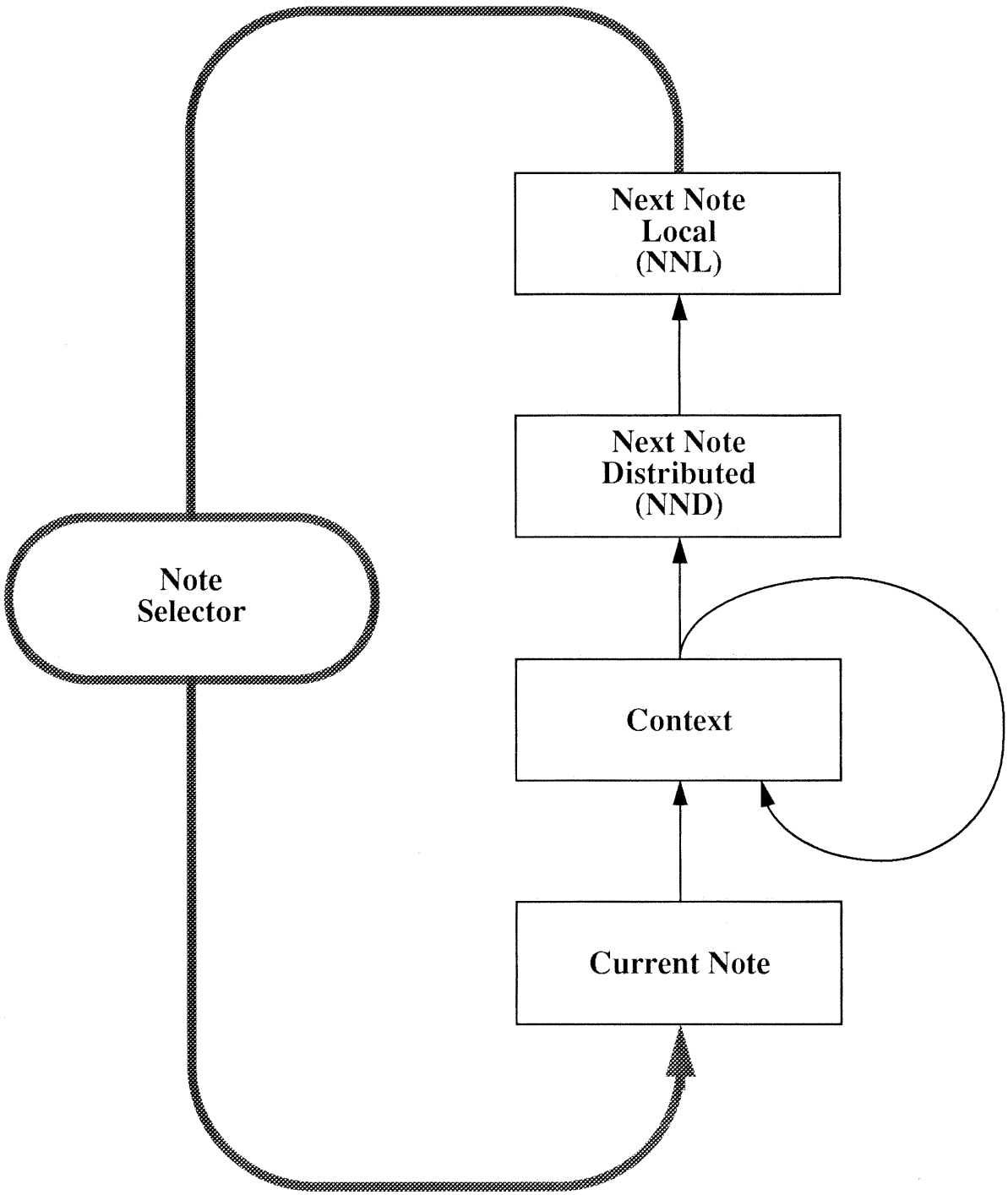


Figure 1. The CONCERT architecture. Rectangles indicate a layer of units, directed lines indicate full connectivity from one layer to another. The selection process is external to CONCERT and is used to choose among the alternatives proposed by the network during composition.

because f is not limited to implementing a stack and the mapping from the context layer to the output is not limited to being a simple look-up table. From myriad possibilities, the training procedure attempts to find a set of connections that are adequate for performing the next-note prediction task. This involves determining which aspects of the input sequence are relevant for making future predictions and constructing the function f appropriately. Subsequently, the context layer will retain only *task-relevant* information. This contrasts with Todd's (1989) work on connectionist composition in which the recurrent context connections are prewired and fixed, which makes the nature of the information Todd's model retains independent of the examples on which it is trained.

Once CONCERT has been trained, it can be run in *composition mode* to create new pieces. This involves first seeding CONCERT with a short sequence of notes, perhaps the initial notes of one of the training examples. From this point on, the output of CONCERT can be fed back to the input, allowing CONCERT to continue generating notes without further external input. Generally, the output of CONCERT does not specify a single note with absolute certainty; instead, the output is a probability distribution over the set of candidates. It is thus necessary to select a particular note in accordance with this distribution. This is the role of the selection process depicted in Figure 1.

Unit activation rules

The activation rule for the context units is

$$c_i(n) = s\left[\sum_j w_{ij} x_j(n) + \sum_j v_{ij} c_j(n-1)\right], \tag{1}$$

where $c_i(n)$ is the activity of context unit i at step n, $x_j(n)$ is the activity of input unit j at step n, w_{ij} is the connection strength from unit j of the input to unit i of the context layer, and v_{ij} is the connection strength from unit j to unit i within the context layer, and s is the standard logistic activation function rescaled to the range (-1,1). Units in the NND layer follow a similar rule:

$$nnd_i(n) = s\left[\sum_j u_{ij} c_j(n)\right],$$

where $nnd_i(n)$ is the activity of NND unit i at step n and u_{ij} is the strength of connection from context unit j to NND unit i.

The transformation from the NND layer to the NNL layer is achieved by first computing the distance between the NND representation, $\mathbf{nnd}(n)$, and the target (distributed) representation of each pitch i, $\boldsymbol{\rho}_i$:

$$d_i = ||\,\mathbf{nnd}(n) - \boldsymbol{\rho}_i\,||\,,$$

where $||\cdot||$ denotes the length of a vector. This distance is an indication of how well the NND representation matches a particular pitch. The activation of the NNL unit corresponding to pitch i, nnl_i, increases as the distance decreases:

$$nnl_i(n) = \frac{e^{-d_i}}{\sum_j e^{-d_j}}.$$

This normalized exponential transform was first proposed by Bridle (1990) and Rumelhart (in preparation). It produces an activity pattern over the NNL units in which each unit has activity in the range (0,1) and the activity of all units sums to 1. Consequently, the NNL activity pattern can be interpreted as a probability distribution — in this case, the probability that the next note has a particular pitch. The distance measure and the exponential function also have a basis in psychological theory (Shepard, 1987), a point I elaborate on shortly.

Training procedure

CONCERT is trained using a variation of the back propagation algorithm (Rumelhart, Hinton, & Williams, 1986). Back propagation is a method for adjusting the connection strengths within CONCERT so that the network can perform the next-note prediction task for a set of training examples. The algorithm requires first defining a measure of the network's performance — of how good a job the network does at predicting each note in each of the training examples. Commonly, a squared difference measure of error is used:

$$E_{lms} = \sum_{p,n,j} (nnl_j(n,p) - \delta(j, t(n,p)))^2 ,$$

where p is an index over pieces in the training set, n an index over notes within a piece, and j an index over units in the NNL layer; $t(n,p)$ is the target pitch for note n of piece p; $\delta(a,b) = 1$ if $a = b$ or 0 otherwise; and the additional p in $nnl_j(n,p)$ specifies the activity for piece p. This measure is minimized when the output of the unit corresponding to the correct prediction is 1 and the output of all other units is 0.

Another performance measure is sensible in the context of output units that have a probabilistic interpretation (Bridle, 1990; Rumelhart, in preparation). Because each NNL unit's output represents the probabilistic expectation of a pitch, performance depends on predicting the appropriate notes with high probability. This suggests the performance measure

$$L = \prod_{p,n} nnl_{t(n,p)}(n,p) ,$$

which is the joint probability of making the correct prediction for all notes of all pieces.[3] Equivalently, a new error measure can be defined based on the logarithm of L,

$$E = -\log L = -\sum_{p,n} \log nnl_{t(n,p)}(n,p) ,$$

because the logarithm is a monotonic function. E is somewhat easier to work with than L.

Back propagation specifies how the weights in the network should be changed to reduce E. This involves computing the gradient of E with respect to the weights in the network: $\partial E/\partial \mathbf{W}$, $\partial E/\partial \mathbf{V}$, and $\partial E/\partial \mathbf{U}$. The first step in this process is computing the gradient with respect to the activity of units in the NND layer, and then propagating this gradient back to the weights in layers below. For the error measure E and the NNL-unit activation rule,

[3] Of course, this interpretation assumes independence of the predictions, which is certainly not true in CONCERT. However, Bridle (1990) provides another justification, somewhat less intuitive, for this performance measure in terms of an information theoretic criterion.

$$\frac{\partial E}{\partial \mathbf{nnd}(n,p)} = \left[\frac{\mathbf{nnd}(n,p) - \boldsymbol{\rho}_{t(n,p)}}{d_{t(n,p)}} - \sum_i nnl_i(n,p) \frac{\mathbf{nnd}(n,p) - \boldsymbol{\rho}_i}{d_i} \right].$$

Back propagation still cannot be used to train CONCERT directly, because CONCERT contains recurrent connections and the algorithm applies only to feedforward networks. Several variations of the algorithm have been proposed for dealing with recurrent networks (Williams & Zipser, in press). I've used the "unfolding in time" procedure of Rumelhart et al. (1986), which transforms a recurrent network into an equivalent feedforward network. The basic trick involves making a copy of the units in the network for each step in the sequence (Figure 2). If the sequence has ten notes, there will be ten copies of the input units, $\mathbf{x}(1) \cdots \mathbf{x}(10)$, as for the other pools of units. Thus, $\mathbf{x}(n)$ refers to a *particular set* of input units, in contrast to the original architecture where $\mathbf{x}(n)$ refers to the input activities *at a particular point in time*. The weights in each copy of the network are set to be equal to the weights in the original architecture. For example, the weights $\mathbf{W}$ connect $\mathbf{x}(n)$ to $\mathbf{c}(n)$ for each n. Consequently, the dynamics of the unfolded network are identical to those of the original network: $\mathbf{x}(n)$ is integrated with $\mathbf{c}(n-1)$ to form $\mathbf{c}(n)$, and so forth. The difference is that the unfolded network is feedforward; that is, activity in Figure 2 flows strictly upwards, whereas in Figure 1 activity flows from the context layer back to itself.

Applying back propagation to the unfolded architecture is therefore straightforward. The error gradient $\partial E / \partial \mathbf{nnd}(n,p)$ is computed at step n. This error is propagated through the copy of the network corresponding to step n, back to the copy corresponding to step $n-1$, and so on. For each copy of the network, this procedure produces a set of suggested weight changes, $\{\Delta\mathbf{W}(i), \Delta\mathbf{V}(i), \Delta\mathbf{U}(i)\}$, where the index i specifies to which step the weight changes correspond. Because there is only one set of underlying weights — the weights in the original network — the weight changes for each step must be summed together to determine the overall change to the underlying weights:

$$\Delta\mathbf{W} = \sum_{i=1}^{n} \Delta\mathbf{W}(i),$$

and similarly for $\Delta\mathbf{V}$ and $\Delta\mathbf{U}$. The actual weight update is performed only after the entire sequence has been presented.[4]

A practical consideration is that in long sequences, the unfolded architecture is deeply layered, and error propagation is computationally expensive. One solution is to simply terminate error propagation after a certain number of steps. However, my simulations were small enough that such a short cut was unnecessary.

Examining the unfolded architecture in Figure 2, one gains a sense of how CONCERT can discover contingencies far apart in time. Consider, for instance, the input at step 1, $\mathbf{x}(1)$, which is linked to the prediction at step n, $\mathbf{nnl}(n)$, via a series of intermediate layers: $\mathbf{c}(1)$, $\mathbf{c}(2)$, $\cdots$, $\mathbf{c}(n)$, and $\mathbf{nnd}(n)$. The weights along this path are adjusted via back propagation so that input note 1, if it has any predictive utility, will influence CONCERT's output at step n. This is ensured by preserving critical aspects of $\mathbf{x}(1)$ in the context layer until step n. If the propagation of error in the network is limited to a certain number of steps (e.g., Elman, 1990), there is no assurance that CONCERT will retain information early in the sequence specifically to make a prediction at a much later step.

---

[4] An unforgivable pun: Rob Goldstone suggested calling CONCERT's training procedure *Bach propagation.*

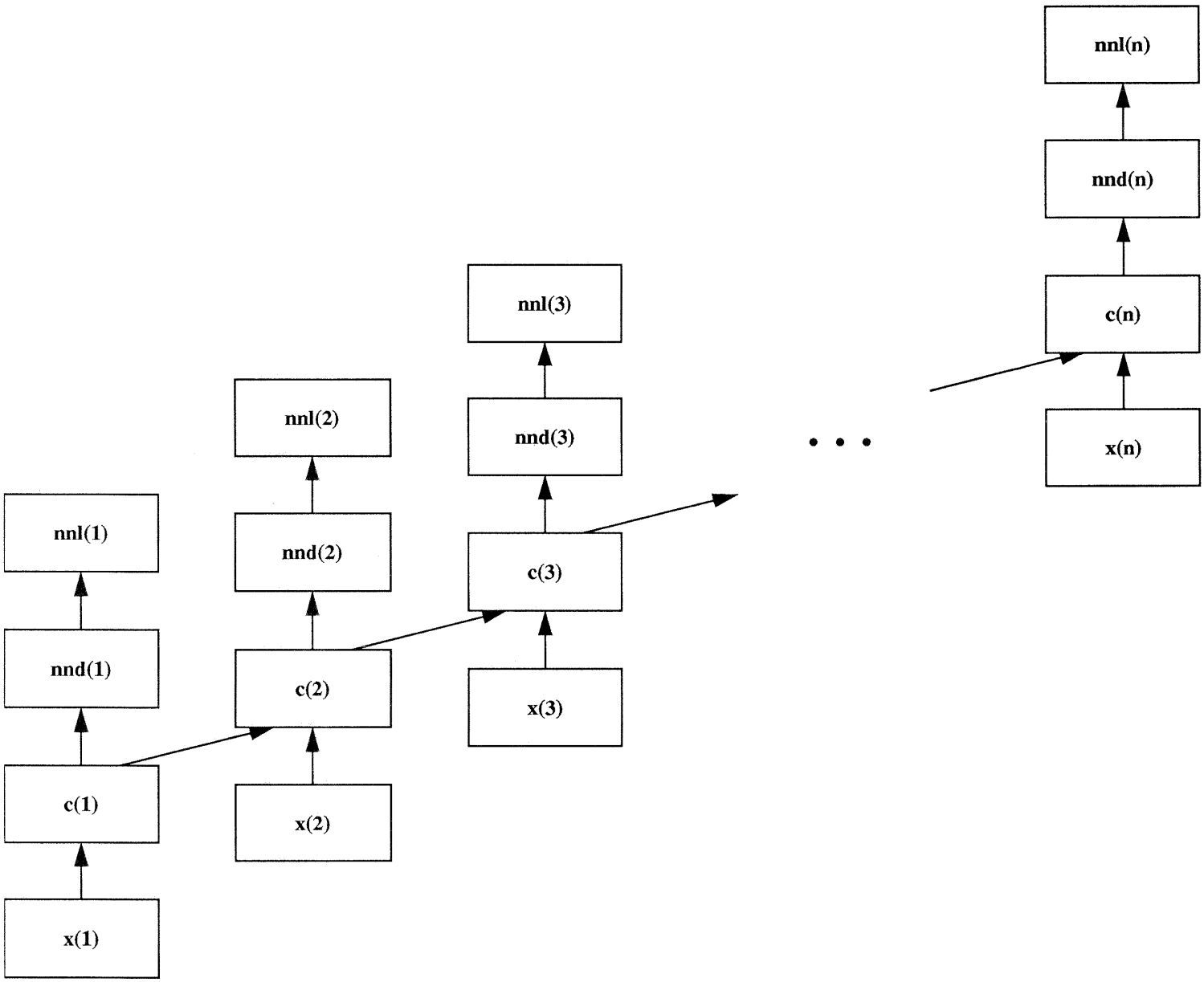


Figure 2. The CONCERT architecture unfolded in time. For each step in the input sequence there is a complete copy of all units in the network. The labels in the boxes indicate the activity vector corresponding to the units: **x** for the input (current note), **c** for the context, **nnd** for the NND units, layer, and **nnl** for the NNL units. The number in parentheses indicates the step.

Capturing higher-order musical organization

In principle, CONCERT should be capable of learning to predict an event from correlated events occurring earlier in the sequence. In my experience, however, back propagation is not sufficiently powerful to discover arbitrary contingencies, in particular those which span long temporal intervals. For example, if a network is trained on sequences in which one event predicts another, the relationship is not hard to learn if the two events are separated by only a few unrelated intervening events, but as the number of intervening events grows, a point is often reached where the relationship cannot be learned (Mozer, 1989).

This presents a serious limitation of using back propagation to induce musical structure because critical structure can be found at long time scales as well as short. A musical piece is more than a linear string of notes. Minimally, a piece should be characterized as a set of musical phrases, each of which is composed of a sequence of notes. Within a phrase, local structure can probably be captured by a transition table, e.g., the fact that the next note is likely to be close in frequency to the current, or that if the past few notes have been in ascending order, the next note is likely to follow this pattern. Across phrases, however, a more global view of the organization is necessary. To illustrate, consider perhaps the simplest phrase structure, AABA. A and B each represent a musical phrase of, say, 20 notes; the piece is thus composed of two repetitions of phrase A, followed by phrase B, followed by a final repetition of A. To correctly predict the third repetition of individual notes in A, it is necessary to remember notes that occurred 40 steps back. Moreover, little of the intervening information is relevant. It is exactly in circumstances such as these that back propagation in time appears to perform poorly. Only the blindest of optimists would claim that the learning procedure for CONCERT just proposed would scale well as the size of the pieces grows and as the amount of global structure increases.

To give a sense of the magnitude of the problem, consider an analogy to analyzing a written text. The task is to predict the next letter in the text based on the previous context. Knowledge of English orthography is one type of information that can be used; for example, a "u" is almost certain to follow a "q". Not only can one letter be predicted from other letters of a word, but one word can be predicted from other words in a sentence, and one sentence from other sentences in a paragraph, etc. To take advantage of the many levels of structure, nothing short of an understanding of orthography, syntax, semantics, *and* pragmatics is required. While orthographic constraints involve local structure — the last couple of letters in the input sequence — other aspects of language demand a more global view of the text.

The difficult problem of learning global as well as local structure hasn't been specifically addressed by connectionist learning theoreticians. I propose a relatively simple first step towards a solution. The basic idea involves building a *reduced description* (Hinton, 1988) of the sequence that makes global aspects more explicit or more readily detectable. In the case of the AABA structure, this might involve taking the sequence of notes composing A and redescribing them simply as "A". Based on this reduced description, recognizing the phrase structure AABA would involve little more than recognizing the sequence AABA. By constructing the reduced description, the problem of detecting global structure has been turned into the simpler problem of detecting local structure.

The challenge of this approach is to devise an appropriate reduced description. Thus far in my work, I've experimented with a crude scheme that has shown some merits. This scheme constructs a reduced description that is a bird's eye view of the musical piece, sacrificing a representation of individual notes for the overall contour of the piece. Imagine playing back a song on a tape recorder at double the regular speed. The notes are to some extent blended together and indistinguishable. However, events at a coarser time scale become more explicit, such as a general ascending trend in pitch or a repeated progression of notes. Figure 3 illustrates the idea. The curve in the top graph, depicting a sequence of

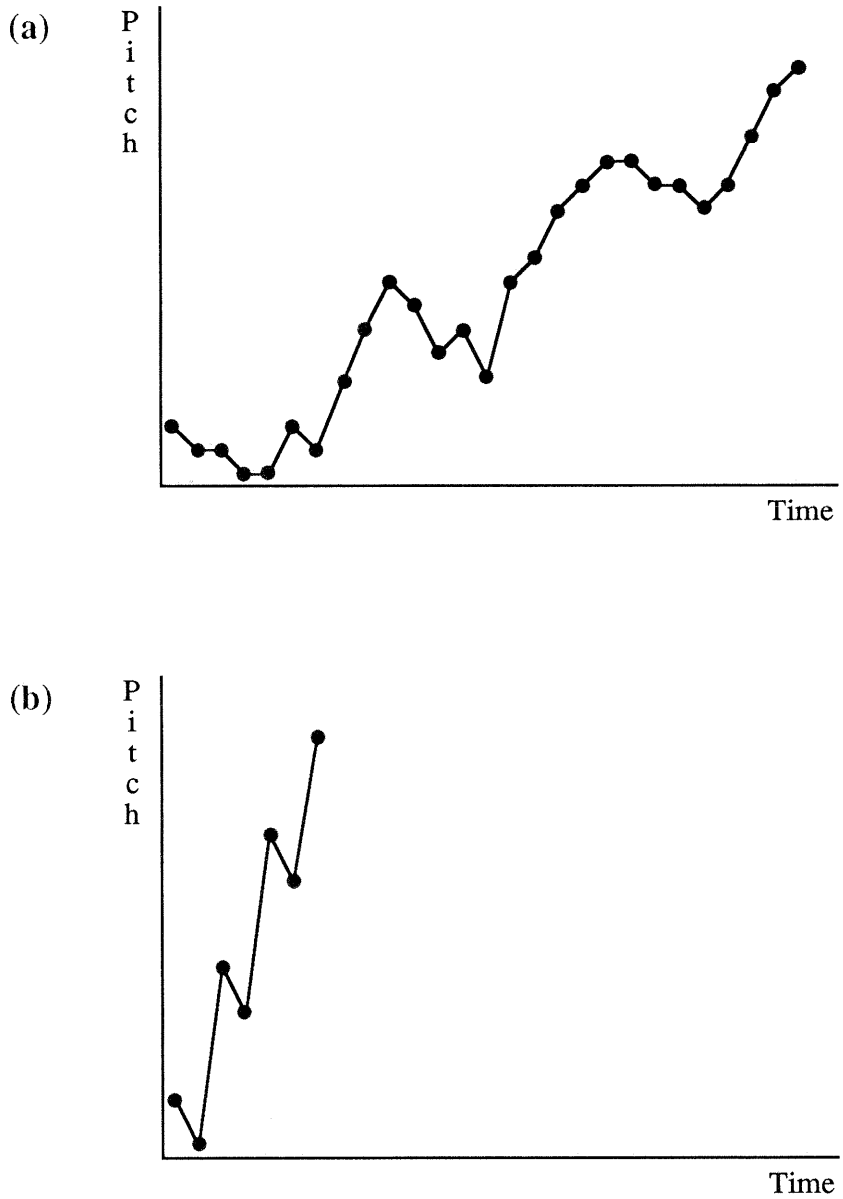


Figure 3. (a) A sequence of individual notes. The vertical axis indicates the pitch, the horizontal axis time. Each point corresponds to a particular note. (b) A smoothed, compact view of the sequence.

individual pitches, has been smoothed and compressed to produce the bottom graph. Mathematically, "smoothed and compressed" means that the waveform has been low-pass filtered and sampled at a lower rate. The result is a waveform in which the alternating upwards and downwards flow is unmistakable.

Multiple views of the sequence are realized in CONCERT using context units that operate with different *time constants*. With a simple modification to the context unit activation rule (Equation 1),

$$c_i(n) = \tau_i c_i(n-1) + (1-\tau_i) s\left[\sum_j w_{ij} x_j(n) + \sum_j v_{ij} c_j(n-1)\right], \tag{2}$$

where each context unit i has an associated time constant, τ_i, that ranges from 0 to 1 and determines the responsiveness of the unit — the rate at which its activity changes. With $\tau_i = 0$, the activation rule reduces to Equation 1 and the unit can sharply change its response based on a new input. With large τ_i, the unit is sluggish, holding on to much of its previous value and thereby averaging the response to the net input over time. At the extreme of $\tau_i = 1$, the second term drops out and the unit's activity becomes fixed. Thus, large τ_i smooth out the response of a context unit over time. This is one property of the waveform in Figure 3b relative to the waveform in Figure 3a.

The other property, the compactness of the waveform, is also achieved by a large τ_i, although somewhat indirectly. The key benefit of the compact waveform in Figure 3b is that it allows a longer period of time to be viewed in a single glance, thereby explicating contingencies occurring during this interval. Equation 2 also facilitates the learning of contingencies over longer periods of time. To see why this is the case, consider the relation between the error derivative with respect to the context units at step n, $\partial E/\partial c(n)$, and the error back propagated to the previous step, $n-1$. One contribution to $\partial E/\partial c_i(n-1)$, from the first term in Equation 2, is

$$\frac{\partial E}{\partial c_i(n)} \frac{\partial}{\partial c_i(n-1)}\left[\tau_i c_i(n-1)\right] = \tau_i \frac{\partial E}{\partial c_i(n)}.$$

This means that when τ_i is large, most of the error signal in context unit i at note n is carried back to note $n-1$. Thus, the back propagated error signal can make contact with points further back in time, facilitating the learning of more global structure in the input sequence.

Several comments regarding this approach.

- Time constants have been incorporated into the activation rules of other connectionist architectures. McClelland's (1979) cascade model makes use of time constants in a feedforward network. The continous-time networks of Pearlmutter (1989) and Pineda (1987) are based on a differential equation update rule, of which Equation 2 is a discrete time version. However, none of this work has exploited time constants to control the temporal responsivity of individual units.
- Although Figure 3 depicts only two time scales, context units can operate at many different time scales, with smaller values of τ_i specializing the units to be sensitive to local properties of the sequence and larger values specializing the units to be more sensitive to global properties.
- Equation 2 suggests one particular type of reduced description, consisting of a smoothed and compressed representation of the context unit response over time. This is a simple-minded reduced description; ideally, one would like the reduced description to characterize meaningful "chunks" or events in the input sequence. Yoshiro Miyata and David Burr (personal communication, 1990) have taken a step in this direction with a two-level hierarchical architecture in which the lower level detects local structure in the input sequence and the higher level detects more

global structure in the output of the lower level.

Pitch representation

Having described CONCERT's architecture and training procedure, I turn to the representation of pitch. To accomodate a variety of music, CONCERT needs the ability to represent a range of about four octaves. Using standard musical notation, these pitches are labeled as follows: `C1`, `D1`, ..., `B1`, `C2`, `D2`, ... `B2`, `C3`, ... `C5`, where `C1` is the lowest pitch and `C5` the highest. Sharps and flats are denoted with `#` and `b`, respectively, e.g., `C#3` and `Gb2`. Within an octave, there are twelve chromatic steps; the range `C1`-`C5` thus includes 49 pitches.

Perhaps the simplest representation of pitch is to have one unit for each possibility. The pitch `C1` would be represented by the activity vector $[1\,0\,0\,\cdots]^T$, `C#1` by the vector $[0\,1\,0\,\cdots]^T$, and so forth. An alternative would be to represent pitch by a single unit whose activity was proportional to the frequency of the pitch. One might argue that the choice of a pitch representation is not critical because back propagation can, in principle, discover an alternative representation well suited to the task (Hinton, 1987). In practice, however, researchers have found that the choice of external representation is a critical determinant of the network's ultimate performance (e.g., Denker et al., 1987; Mozer, 1987). Quite simply, the more task-appropriate information that is built into the network, the easier the job the learning algorithm has.

Laden and Keefe (1989) advocate the approach of including as much information as possible from psychoacoustics into the design of networks for music perception and cognition. They have developed a model of chord classification that categorizes triads as major, minor, or diminished chords. Classification performance is superior with the use of a representation that explicitly encodes harmonics of the fundamental pitches.

In accord with this approach, and because I am asking the network to make predictions about melodies that *people* have composed or to generate melodies that *people* perceive as pleasant, a central ingredient of my work has been to furnish CONCERT with a psychologically-motivated representation of pitch. By this, I mean that notes that people judge to be similar should have similar representations in the network, indicating that the representation in the head matches the representation in the network. The local representation scheme proposed earlier clearly does not meet this criterion. In the local represention, every pair of pitches is equally similar (using either the distance or angle between vectors as a measure of similarity), yet people perceive pairs of notes like `C1` and `C#1` to be more similar than, say, `C1` and `A4`. Other obvious representations of pitch do not meet the criterion either. For example, a direct encoding of frequency does not capture the similarity that people hear between octaves.

Shepard (1982) has systematically studied the similarity of pitches by asking people to judge the perceived similarity of pairs of pitches. He has proposed a theory of generalization (Shepard, 1987) in which the similarity of two items is exponentially related to their distance in an internal or "psychological" representational space.[5] For the internal representation of pitch, Shepard has proposed a five-dimensional space, depicted in Figure 4. In this space, each pitch specifies a point along the *pitch height* (or *PH*) dimension, an (x,y) coordinate on the *chromatic circle* (or *CC*), and an (x,y) coordinate on the *circle of fifths* (or *CF*). I will refer to this representation as PHCCCF, after its three components. The pitch height component specifies the logarithm of the frequency of a pitch; this logarithmic transform places tonal half-steps at equal spacing from one another along the pitch height axis. In the chromatic

[5] This is one justification for the exponential function in the NNL layer.

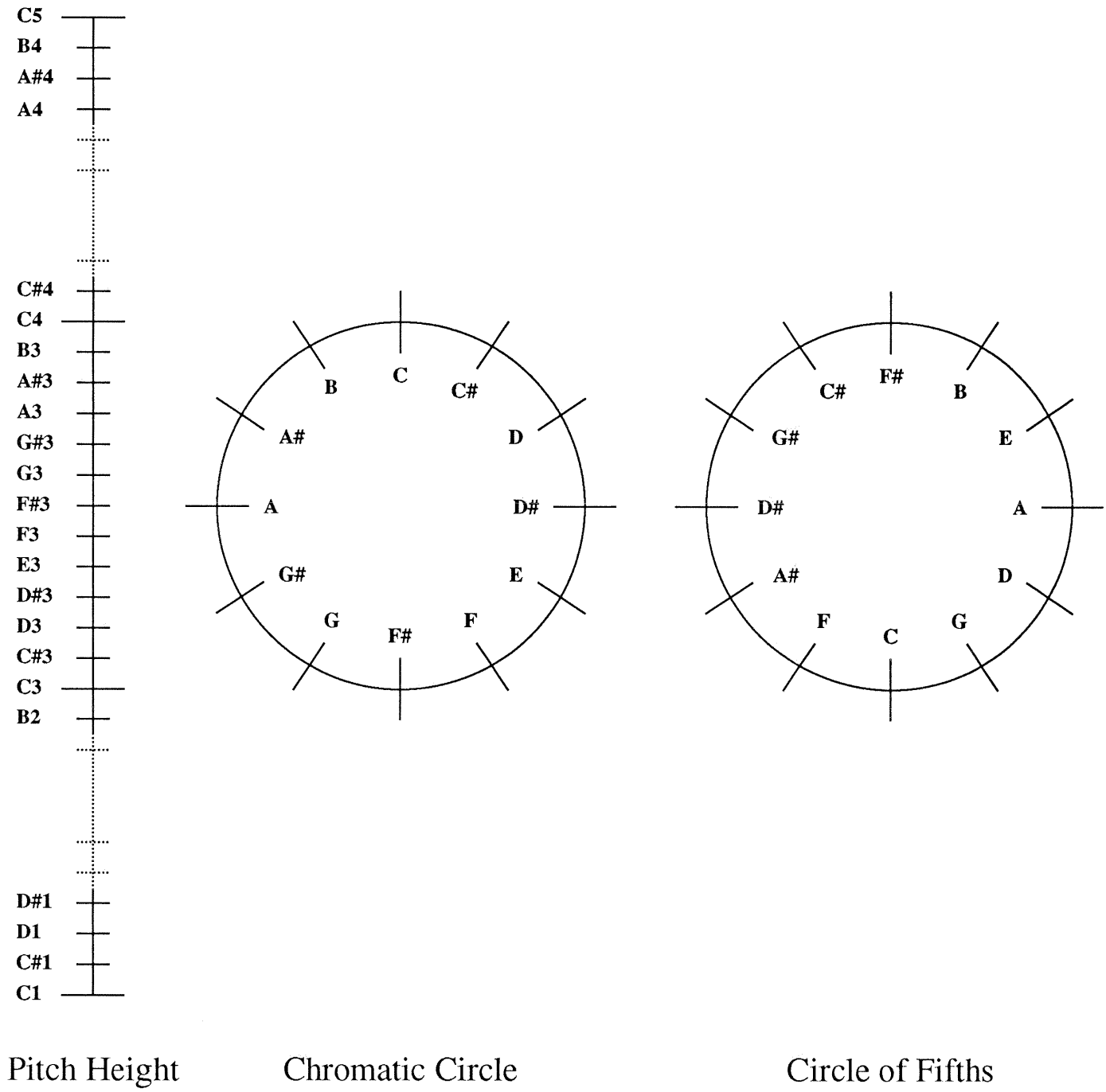


Figure 4. Shepard's (1982) pitch representation.

circle, neighboring pitches are a tonal half-step apart. In the circle of fifths, the perfect fifth of a pitch is the next pitch immediately counterclockwise.[6] The proximity of two pitches in the five-dimensional PHCCCF space can be determined simply by computing the Euclidean distance between their representations.

Shepard substantiates the psychological validity of the PHCCCF representation in detail. I will briefly point out some of its benefits. Consider first the PH and CC components. In this three-dimensional subspace, pitches form a helix in which the winding of the helix is due to the chromatic circle and the height is due to the pitch height. As pitches proceed up the chromatic scale, they wind up the helix. Pitches exactly one octave apart are directly above one another on the helix; that is, they have the same locus on the chromatic circle but different values of pitch height. For this reason, octaves have similar representations. Depending on how the PH component is scaled relative to the CC (i.e., how elongated the helix is), pitches like `C1` and `C2` may even be closer in the representational space than pitches like `C1` and `B1`, even though `C1` is closer to `B1` in frequency.

The circle of fifths endows the representation with other desirable properties. First, the circle localizes the tones in a musical key. Any seven adjacent tones correspond to a particular key. For instance, the tones of the C major and A minor diatonic scales — `C`, `D`, `E`, `F`, `G`, `A`, and `B` — are grouped together on the circle of fifths. The most common pentatonic keys are similarly localized. Second, and perhaps more critical, the circle of fifths can explain the subjective equality of the intervals of the diatonic scale. To elaborate, Shepard points out that people tend to hear the successive steps of the major scale as equivalent, although with respect to log frequency, some of the intervals are only half as large as others. For example, in C major, the `E-F` and `B-C` steps are half tones apart (minor seconds) while all others are a whole tone apart (major seconds). The combination of the pitch height and circle of fifths permits a representation in which the distance between all major and minor seconds is the same. This is achieved by using a scale ratio of approximately 3:1 for the chromatic circle relative to the circle of fifths.

One desireable property of the overall PHCCCF representation is that distances between pitches are invariant under transposition. Consider any two pitches, say, `D2` and `G#4`. Transposing the pitches preserves the distance between them in the PHCCCF representation. Thus, the distance from `D2` to `G#4` is the same as from `E2` to `A#4`, from `D1` to `G#3`, and so forth. See Bharucha (in press) for a further discussion of the psychological issues involved in the representation of musical pitch.

The relative importance of the PH, CC, and CF components can be varied by adjusting the diameters of the chromatic circle and circle of fifths. For example, if the two circles have the same diameter, then, in terms of the CC and CF components, the distance between `C` and `G` is the same as the distance between `C` and `B`. This is because `B` is one notch from the `C` on the chromatic circle and five notches on the circle of fifths, while the `G` is five notches away on the chromatic circle and one on the circle of fifths. However, if the diameter of the chromatic circle is increased, then `C` is closer to `B` than to `G` (based on the distance in the four-dimensional CC and CF subspace); if the diameter is decreased, `C` is closer to `G` than to `B`. If the diameters of both circles are decreased relative to the pitch height scale, then pitch frequency becomes the most important determinant of similarity. Shepard argues that the weighting of the various components depends on the particular musical task and the listener's expertise. Based on Shepard's evidence, a reasonable representation for expert musicians is to weigh the CF and CC components equally, and to set the diameter of the CC and CF components equal to the distance of one octave in PH. This is the scale shown in Figure 4.

---

[6] The perfect fifth is a musically significant interval. The frequency ratio of a note to its perfect fifth is 2:3, just as the frequency ratio of a note to its octave is 1:2.

The final issue to discuss is how the PHCCCF representation translates into an activity vector over a set of connectionist units. A straightforward scheme is to use five units, one for pitch height and two pairs to encode the (x,y) coordinates of the pitch on the two circles. One problem with this scheme is that, if the units have the usual sigmoidal activation function, equal spacing of tones in pitch height or on the circles in unit *activity* space is not preserved in unit *net input* space. This means that context units attempting to activate NND units do not reap the full benefit of the representation (e.g., transposition invariance). A second problem with the simple five-unit scheme is that the activity of each unit encodes a coordinate value directly; there are 7 discrete values for the x- and y-coordinates of the circles, 49 for the pitch height. Consequently, minor perturbations of the activity vector could lead to misinterpretations.

Due to these problems, I have opted for an alternative representation of the CC and CF components. The representation involves 6 binary-valued units to represent a tone on each circle; the representation for chromatic circle tones is shown in Table 2. This representation preserves the essential distance relationships among tones on the chromatic circle: the distance between two tones is monotonically related to the angle between the tones. Because each unit has to encode only two distinct values, the representation is less sensitive to noise than is one in which each unit encodes a real value.

Unfortunately, I do not believe there is a similar scheme that can be used to encode pitch height in a boolean space of reasonably low dimensionality that preserves intrinsic distance relationships. Consequently, I have stayed with a single linear unit for pitch height. Its activity is scaled to range from -9.798 for `C1` to +9.798 for `C5`. This scaling achieves the desired property previously described that the distance in the CC or CF component between pitches on opposite sides of the circle equals the distance between pitches one octave apart in the PH component.[7]

The PHCCCF representation consists of 13 units altogether. Sample activity patterns for some pitches are shown in Table 3. Rests (silence) are assigned a code, listed in the last row of the Table, that distinguish them from all pitches. The end of a piece is coded by a series of rests.

Table 2: Representation of tones on chromatic circle

tone	*representation*					
C	−1	−1	−1	−1	−1	−1
C#	−1	−1	−1	−1	−1	+1
D	−1	−1	−1	−1	+1	+1
D#	−1	−1	−1	+1	+1	+1
E	−1	−1	+1	+1	+1	+1
F	−1	+1	+1	+1	+1	+1
F#	+1	+1	+1	+1	+1	+1
G	+1	+1	+1	+1	+1	−1
G#	+1	+1	+1	+1	−1	−1
A	+1	+1	+1	−1	−1	−1
A#	+1	+1	−1	−1	−1	−1
B	+1	−1	−1	−1	−1	−1

[7] Although a PH scale factor of 9.798 was used for the target NND representation, ρ_i, a PH scale factor of 1.0 was used for the input representation. This was based on empirical studies of what scale factors yielded the best performance. The primary reason that a PH scale factor other than 1.0 on the inputs causes difficulties is that gradient descent in the error surface becomes messier when different units have different activity ranges (Widrow & Stearns, 1985).

Table 3: PHCCCF representation for selected pitches

pitch	*PH*	*CC*						*CF*					
C1	–9.798	+1	+1	+1	–1	–1	–1	–1	–1	–1	+1	+1	+1
F#1	–7.349	–1	–1	–1	+1	+1	+1	+1	+1	+1	–1	–1	–1
G2	–2.041	–1	–1	–1	–1	+1	+1	–1	–1	–1	–1	+1	+1
C3	0	+1	+1	+1	–1	–1	–1	–1	–1	–1	+1	+1	+1
D#3	1.225	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
E3	1.633	–1	+1	+1	+1	+1	+1	+1	–1	–1	–1	–1	–1
A4	8.573	–1	–1	–1	–1	–1	–1	–1	–1	–1	–1	–1	–1
C5	9.798	+1	+1	+1	–1	–1	–1	–1	–1	–1	+1	+1	+1
rest	0	+1	–1	+1	–1	+1	–1	+1	–1	+1	–1	+1	–1

As with any distributed representation, there are limitations as to how many and which pitches can be represented simultaneously. The issue arises because the NND layer needs to be able to encode a set of alternatives, not just a single pitch. If, say, A1, D2, and E2 are equally likely as the next note, the NND layer must indicate all three possibilities. To do so, it must produce an activity vector that is nearer to $\boldsymbol{\rho}_{A1}$, $\boldsymbol{\rho}_{D2}$, and $\boldsymbol{\rho}_{E2}$ than to other possibilities. The point in PHCCCF space that is simultaneously closest to the three pitches is simply the average vector, $(\boldsymbol{\rho}_{A1}+\boldsymbol{\rho}_{D2}+\boldsymbol{\rho}_{E2})/3$. Table 4 shows the pitches nearest to the average vector. As hoped for, A1, D2, and E2 are the nearest three. This is not always the case, though. Table 5 shows the pitches nearest to the average vector which represents the set {A1, D2, D#2}. This illustrates the fact that certain clusters of pitches are more compact in the PHCCCF space than others. The PHCCCF representation not only introduces a similarity structure over the pitches, but also a limit on the combinations of pitches that can be considered simultaneously. Arbitrary limitations are a bad thing in general, but here, the limitations are *theoretically motivated.*[8]

One serious shortcoming of the PHCCCF representation is that it is based on the similarity between pairs of notes presented in isolation. Listeners of music do not process individual notes in isolation; notes appear in a musical context which suggests a musical key which in turn contributes to an interpretation of the note. Some psychologically-motivated work has considered the effects of context or musical key on pitch representation (Krumhansl, 1990; Krumhansl & Kessler, 1982; Longuet-Higgins, 1976, 1979). I

Table 4: Distance from representation of {A1,D2,E2} to nearest 10 pitches

rank	*pitch*	*distance*	*rank*	*pitch*	*distance*
1	D2	2.528	6	C#2	4.422
2	E2	2.779	7	A2	4.422
3	A1	3.399	8	E1	4.441
4	B1	3.859	9	G1	4.497
5	C2	4.130	10	G2	4.497

Table 5: Distance from representation of {A1,D2,D#2} to nearest 10 pitches

rank	*pitch*	*distance*	*rank*	*pitch*	*distance*
1	D2	2.373	6	D#2	3.774
2	C2	3.277	7	A1	3.946
3	E2	3.538	8	F2	4.057
4	C#2	3.654	9	A#1	4.146
5	B1	3.714	10	G1	4.323

[8] I hope this isn't too reminiscent of the old saying, "It's not a bug; it's a feature."

believe that CONCERT could be improved considerably by incorporating the ideas in this work. Fortunately, it does not require discarding the PHCCCF representation altogether, because the PHCCCF representation shares many properties in common with the representations suggested by Krumhansl and Kessler and by Longuet-Higgens.

Simulation Experiments

Extending a C major diatonic scale

To start with a simple experiment, CONCERT was trained on a single sequence consisting of three octaves of a C major diatonic scale: `C1 D1 E1 F1 ··· B3`. The target at each step was the next note in the scale: `D1 E1 F1 G1 ··· C4`. CONCERT is said to have learned the sequence when, at each step, the activity of the NNL unit representing the target at that step is more active than any other NNL unit. In 10 replications of the simulation with different initial random weights, 15 context units, a learning rate of .005, and no momentum, CONCERT learned the sequence in about 30 passes. Following training, CONCERT was tested on four octaves of the scale. CONCERT correctly extended its predictions to the fourth octave, except that in 4 of the 10 replications, the final note, `C5`, was transposed down an octave. Table 6 shows the CONCERT's output for two octaves of the scale. Octave 3 was part of the training sequence, but octave 4 was not. Activities of the three most active output units are shown. Because the output activities can be interpreted as probabilities, one can see that the target is selected with high confidence.

CONCERT was able to learn the training set with as few as 2 context units, although surprisingly, generalization performance tended to improve as the number of context units was increased. CONCERT was also able to generalize from a 2 octave training sequence, but it often transposed notes down an octave.

Learning the structure of diatonic scales

In this simulation, I trained CONCERT on a set of diatonic scales in various keys over a one octave range, e.g., `D1 E1 F#1 G1 A1 B1 C#2 D2`. Thirty-seven such scales can be made using pitches in the `C1-C5` range. The training set consisted of 28 scales — roughly 75% of the corpus — selected at random, and the test set consisted of the remaining 9. In 10 replications of the simulation using 20 context units, CONCERT mastered the training set in approximately 55 passes. Generalization performance

Table 6: Performance on octaves 3 and 4 of C major diatonic scale

input pitch	*output unit activities*					
C3	D3	0.961	C3	0.017	E3	0.014
D3	E3	0.972	D3	0.012	F3	0.007
E3	F3	0.982	D#3	0.008	G3	0.006
F3	G3	0.963	F3	0.015	A3	0.010
G3	A3	0.961	G3	0.024	B3	0.012
A3	B3	0.972	A3	0.025	C4	0.002
B3	C4	0.979	A#3	0.010	C#4	0.005
C4	D4	0.939	C4	0.040	E4	0.009
D4	E4	0.968	D4	0.018	F4	0.006
E4	F4	0.971	D#4	0.016	E4	0.005
F4	G4	0.931	F4	0.037	F#4	0.015
G4	A4	0.938	G4	0.044	B4	0.007
A4	B4	0.915	A4	0.080	A#4	0.003
B4	C5	0.946	A#4	0.040	B4	0.011

was tested by presenting the scales in the test set one note at a time and examining CONCERT's prediction. This is not the same as running CONCERT in composition mode because CONCERT's output was not fed back to the input; instead, the input was a predetermined sequence. Of the 63 notes to be predicted in the test set, CONCERT achieved remarkable performance: 98.4% correct. The few errors were caused by transposing notes one full octave or one tonal half step.

To compare CONCERT with a transition table approach, I built a second-order transition table from the training set data and measured its performance on the test set. The transition table prediction (i.e., the note with highest probability) was correct only 26.6% of the time. The transition table is somewhat of a straw man in this environment: A transition table that is based on absolute pitches is simply unable to generalize correctly. Even if the transition table encoded relative pitches, a third-order table would be required to master the environment. Kohonen's musical grammar faces the same difficulties as a transition table.

Learning random walk sequences

In this simulation, I generated ten-element sequences according to a simple rule: The first pitch was selected at random, and then successive pitches were either one step up or down the C major scale from the previous pitch, the direction chosen at random. The pitch transitions can easily be described by a transition table, as illustrated in Table 1. CONCERT, with 15 context units, was trained for 50 passes through a set of 100 such sequences. If CONCERT has correctly inferred the underlying rule, its predictions should reflect the plausible alternatives at each point in a sequence. To test this, a set of 100 novel random walk sequences was presented. After each note n of a sequence, CONCERT's performance was evaluated by matching the top two predictions — the two pitches with highest activity — against the actual note $n+1$ of the sequence. If note $n+1$ was not one of the top two predictions, the prediction was considered to be erroneous. In ten replications of the simulation, the mean performance was 99.95% correct. Thus, CONCERT was clearly able to infer the structure present in the patterns. CONCERT performed equally well, if not better, on random walks in which chromatic steps (up or down a tonal half step) were taken.

Learning interspersed random walk sequences

The sequences in this simulation were generated by interspersing the elements of two simple random walk sequences of the sort just described. Each interspersed sequence had the following form: a_1, b_1, a_2, b_2, $\cdots$, a_5, b_5, where a_1 and b_1 are randomly selected pitches, a_{i+1} is one step up or down from a_i on the C major scale, and likewise for b_{i+1} and b_i. Each sequence consisted of ten notes. CONCERT, with 25 context units, was trained on 50 passes through a set of 200 examples and was then tested on an additional 100. In contrast to the simple random walk sequences, it is impossible to predict the second note in the interspersed sequences (b_1) from the first (a_1). Thus, this prediction was ignored for the purpose of evaluating CONCERT's performance. CONCERT achieved a performance of 91.7% correct. About half the errors were ones in which CONCERT transposed a correct prediction by an octave. Excluding these errors, performance improved to 95.8% correct.

To capture the structure in this environment, a transition table approach would need to consider at least the previous two notes. However, such a transition table is not likely to generalize well because, if it is to be assured of predicting a note at step n correctly, it must observe the note at step $n-2$ in the context of *every possible* note at step $n-1$. I constructed a second-order transition table from CONCERT's training set. Using a testing criterion analogous to that used to evaluate CONCERT, the transition table achieved a performance level on the test set of only 67.1% correct. Kohonen's musical grammar would face the same difficulty as the transition table in this environment.

Learning AABA phrase patterns

The melodies in this simulation were formed by generating two random walk phrases, call them A and B, and concatenating the phrases in an AABA pattern. The A and B phrases consisted of five-note ascending or descending chromatic scales, respectively, the first note selected at random. The complete melody then consisted of 21 elements — four phrases of five notes followed by a rest marker — an example of which is:

```
F#2 G2 G#2 A2 A#2 F#2 G2 G#2 A2 A#2 C4 B3 A#3 A3 G#3 F#2 G2
G#2 A2 A#2 REST.
```

These melodies are simple examples of sequences that have both local and global structure. The local structure is derived from the relations among notes within a phrase, the global structure is derived from the relations among phrases. This environment was designed to examine: (1) how well CONCERT could cope with multiple levels of structure and (2) the utility of varying the rate of temporal integration of the context units (the parameter τ in Equation 2).

Two versions of CONCERT were tested, each with 35 context units. In the *standard* version, all 35 units had $\tau = 0$; in the *reduced description* or *RD* version, 30 had $\tau = 0$ and 5 had $\tau = .8$. The training set consisted of 200 examples and the test set another 100 examples. Ten replications of each simulation were run for 300 passes through the training set.

Because of the way that the sequences were organized, certain notes could be predicted based on local structure whereas other notes required a more global memory of the sequence. In particular, the second through fifth notes within a phrase could be predicted based on knowledge of the immediately preceeding note.[9] To predict the first note in the repeated A phrases and to predict the rest at the end of a sequence, more global information is necessary. Thus, the analysis, summarized in Table 7, was split to distinguish between notes that required only local structure and notes that required more global structure. Performance requiring global structure was significantly better for the RD version ($F(1,9)=20.7$, $p<.001$), but there was no reliable difference for performance requiring only local structure ($F(1,9)<1$). A large part of the performance enhancement for the RD version on global structure was due to its improved ability at predicting the rest at the end of a sequence: 62.2% for the standard version versus 82.6% for the RD version ($F(1,9)=32.4$, $p<.001$). However, there was also a modest improvement in the ability to predict the first note of the repeated A phrases: 71.3% for the standard version versus 75.1% for the RD version ($F(1,9)=4.85$, $p=.055$).

Table 7: Performance on AABA phrases

structure	standard version	RD version
local	92.2%	92.1%
global	68.3%	77.6%

[9] This is not quite correct because the A phrases were ascending and the B phrases descending. Thus, to predict the second note of a phrase it is also necessary to know whether the current phrase is A or B, a global property of the sequence. However, for the third through fifth notes in a phrase, the prediction can be based solely on the two immediately preceeding notes — local information.

Experiments with different values of τ in the range .7–.95 yielded qualitatively similar results, as did experiments in which the A and B phrases were formed by random walks in the key of C. Overall, modest improvements in performance are observed, yet it is somewhat disappointing that the global structure is never learned as well as the local.

Generating new melodies in the style of Bach

In a final experiment, I trained CONCERT on the melody line of a set of ten simple piano pieces by J. S. Bach (Table 8). The set of pieces is not particularly coherent; it includes a variety of musical styles. The primary thing that the pieces have in common is their composer. The pieces had several voices, but the melody generally appeared in the treble voice. Importantly, to naive listeners the extracted melodies sounded pleasant and coherent without the accompaniment.

In the training data, each piece was terminated with a rest marker (the only rests in the pieces). This allowed CONCERT to learn not only the notes within a piece but also when the end of the piece was reached. Further, each major piece was transposed to the key of C major and each minor piece to the key of A minor. This was done to facilitate learning because the pitch representation does not take into account the notion of musical key; hopefully, a more sophisticated pitch representation would avoid the necessity of this step.

In this simulation, each note was represented by a duration as well as a pitch. The duration representation consisted of five units and was somewhat analogous the PHCCCF representation for pitch. It allowed for the representation of sixteenth, eighth, quarter, and half notes, as well as triplets. Also included in this simulation were two additional input ones. One indicated whether the piece was in a major versus minor key, the other indicated whether the piece was in 3/4 meter versus 2/4 or 4/4. These inputs were fixed for a given piece.

Learning the examples involves predicting a total of 1,260 notes altogether, no small feat. CONCERT was trained with 40 hidden units, 35 with $\tau = 0$ and 5 with $\tau = 8$, for 3000 passes through the training set. The learning rate was gradually lowered from .0004 to .0002. By the completion of training, CONCERT could correctly predict about 95% of the pitches and 95% of the durations correctly. New pieces can be created by presenting a few notes to start and then running CONCERT in composition mode. Two examples of compositions produced by CONCERT are shown in Figure 5 The first four notes of each composition, used to seed CONCERT, are from one of the training examples. CONCERT specifies the end of a composition by producing a rest marker. The compositions often contain brief excerpts from the training examples. This is because CONCERT has learned the training examples so well that in many contexts, it produces a prediction of one note with probability 1. This means that the selection process does not have

Table 8: Bach training examples

piece	*number of notes*
Minuet in G major (no. 1)	126
Minuet in G major (no. 2)	166
Minuet in D minor	70
Minuet in A minor	84
Minuet in C minor	80
March in G major	153
March in D major	122
March in Eb major	190
Musette in D major	128
Little prelude in C major	121

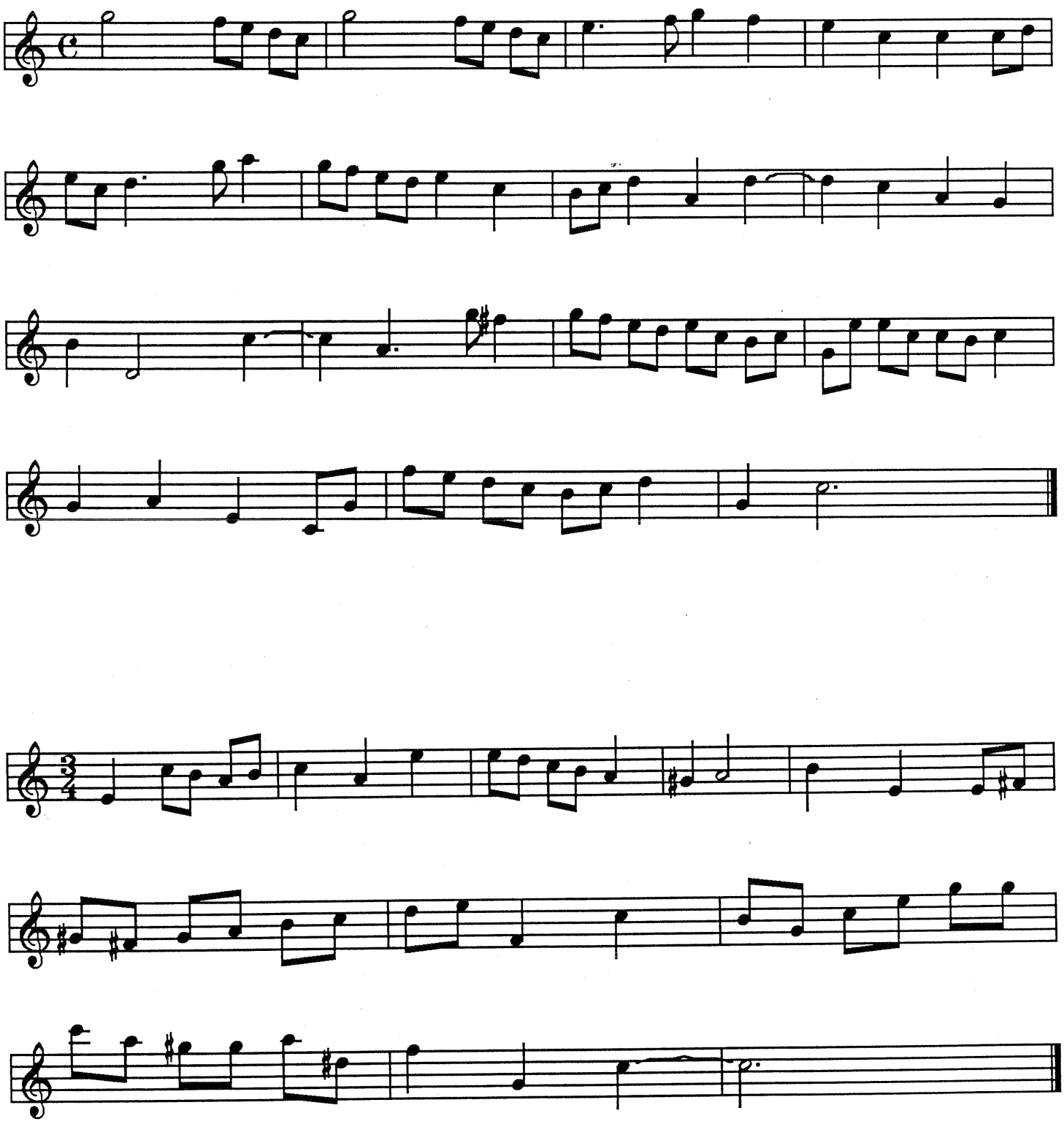


Figure 5. Two sample compositions produced by CONCERT based on the Bach training set.

the opportunity to follow an alternative direction.

The primary deficiency of CONCERT's compositions is that they are lacking in global coherence. A composition might flip back and forth between C major and A minor, or it might incorporate aspects of both the marches and the minuets. Because the shorter compositions have less chance of wandering in this manner, they tend to be more agreeable.

Discussion

Initial results from CONCERT are encouraging. CONCERT is able to learn musical structure of varying complexity, from simple random walk sequences to Bach pieces containing nearly 200 notes. I presented two examples of structure that CONCERT can learn but that cannot be captured by a simple transition table or by Kohonen's musical grammar. One example involved diatonic scales in various keys, the other involved interspersed random walks.

I have motivated CONCERT's architecture primarily from psychological and computational perspectives, but have yet to provide empirical support for this architecture over other possibilities. Is the PHCCCF representation warranted, or would a simpler, perhaps even a local, representation of pitch suffice? Is the NNL layer and the log likelihood performance measure necessary, or could the NNL layer be eliminated and the error be computed directly by comparing the NND activity pattern with the target pitch representation? These questions need to be answered systematically, but informal experiments with alternative architectures and representations have convinced me that CONCERT's performance is remarkably good. These experiments included: (1) the use of localist pitch representations, (2) variants in the PHCCCF representation, such as using two units to represent the circles instead of six, and (3) alternative error measures, such as computing the sequared difference between the NND and target activity patterns. Each variant yielded significantly poorer results in many of the simulations.

Beyond a more systematic examination of alternative architectures, work on CONCERT is heading in three directions. First, the pitch representation is being expanded to account for the perceptual effects of musical context and musical key. Second, CONCERT is being elaborated to include a representation of note duration as well as pitch. The duration representation, as the PHCCCF representation, is based on characteristics of human perception. Preliminary experiments using this duration representation were reported in the Bach simulation. Third, CONCERT is being extended to better handle the processing of global structure in music using the idea of context units that operate at several different temporal resolutions simultaneously. The experiments described in this report have shown modest success using this technique, but much further work remains.

REFERENCES

Bharucha, J. J. (1991). Pitch, harmony, and neural nets: A psychological perspective. In P. M. Todd & D. G. Loy (Eds.), *Connectionism and music*. Cambridge, MA: MIT Press/Bradford Books.

Bridle, J. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 211–217). San Mateo, CA: Morgan Kaufmann.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., & Hopfield, J. (1987). Automatic learning, rule extraction, and generalization. *Complex Systems, 1*, 877–922.

Dodge, C., & Jerse, T. A. (1985). *Computer music: Synthesis, composition, and performance*. New York: Shirmer Books.

Dolson, M. (1989). Machine Tongues XII: Neural networks. *Computer Music Journal, 13*, 28–40.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*, 179–212.

Hinton, G. (1987). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 1–12). Hillsdale, NJ: Erlbaum.

Hinton, G. E. (1988). Representing part-whole hierarchies in connectionist networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 48–54). Hillsdale, NJ: Erlbaum.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume I: Foundations* (pp. 77–109). Cambridge, MA: MIT Press/Bradford Books.

Jones, K. (1981). Compositional applications of stochastic processes. *Computer Music Journal, 5*, 45–61.

Kohonen, T. (1989). A self-learning musical grammar, or "Associative memory of the second kind". *Proceedings of the 1989 International Joint Conference on Neural Networks, 1*, 1–5.

Kohonen, T., Laine, P., Tiits, K., & Torkkola, K. (1991). A nonheuristic automatic composing method. In P. M. Todd & D. G. Loy (Eds.), *Connectionism and music*. Cambridge, MA: MIT Press/Bradford Books.

Krumhansl, C. L. (1990). *Cognitive foundations of musical pitch*. New York: Oxford University Press.

Krumhansl, C. L., & Kessler, E. J. (1982). Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review, 89*, 334–368.

Laden, B., & Keefe, D. H. (1989). The representation of pitch in a neural net model of chord classification. *Computer Music Journal, 13*, 12–26.

Longuet-Higgins, H. C. (1976). Perception of melodies. *Nature, 263*, 646–653.

Longuet-Higgins, H. C. (1979). The perception of music (Review Lecture). *Proceedings of the Royal Society of London, 205B*, 307–332.

Lorrain, D. (1980). A panoply of stochastic 'cannons'. *Computer Music Journal, 3*, 48–55.

Loy, D. G. (1991). Connectionism and musiconomy. In P. M. Todd & D. G. Loy (Eds.), *Connectionism and music*. Cambridge, MA: MIT Press/Bradford Books.

McClelland, J. L. (1979). On the time relations of mental processes: An examination of systems of processes in cascade. *Psychological Review, 86*, 287–330.

Mozer, M. C. (1987). RAMBOT: A connectionist expert system that learns by example. In M. Caudill & C. Butler (Eds.), *Proceedings of the IEEE First Annual International Conference on Neural Networks* (pp. 693–700). San Diego, CA: IEEE Publishing Services.

Mozer, M. C. (1989). A focused back-propagation algorithm for temporal pattern recognition. *Complex Systems, 3*, 349–381.

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation, 1*, 263–269.

Pineda, F. (1987). Generalization of back propagation to recurrent neural networks. *Physical Review Letters, 19*, 2229–2232.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume I: Foundations* (pp. 318–362). Cambridge, MA: MIT Press/Bradford Books.

Rumelhart, D. E. (in preparation). *Connectionist processing and learning as statistical inference.* Hillsdale, NJ: Erlbaum.

Shepard, R. N. (1982). Geometrical approximations to the structure of musical pitch. *Psychological Review, 89*, 305–333.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science, 237*, 1317–1323.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral & Brain Sciences, 11*, 1–74.

Todd, P. M. (1989). A connectionist approach to algorithmic composition. *Computer Music Journal, 13*, 27–43.

Widrow, B., & Stearns, S. D. (1985). *Adaptive signal processing.* Englewood Cliffs, NJ: Prentice-Hall.

Williams, R. J., & Zipser, D. (In Press). Gradient-based learning algorithms for recurrent connectionist networks. In Y. Chauvin & D. E. Rumelhart (Eds.), *Backpropagation: Theory, architectures, and*

applications. Hillsdale, NJ: Erlbaum.